4\ PRTS

VIDEO CODING METHOD OF EXPLOITING THE TEMPORAL REDUNDANCY BETWEEN SUCCESSIVE FRAMES

The present invention relates to a video coding method of exploiting the temporal redundancy between successive frames in a video sequence.

Efficiently encoding video or moving pictures relies heavily on exploiting the temporal redundancy between successive frames in a sequence. Based on the assumption that local motions are slow with respect to the temporal sampling period, several techniques have been proposed for efficiently removing this redundancy. The most successful and acclaimed method is block-based motion prediction, heavily used in nowadays standards such as MPEG4 and H.26L. Roughly speaking, these compression scheme predict a frame in the sequence based on the knowledge of previous frames. The current frame (or predicted) is cut into blocks of fixed size and the best matching block is searched in the reference frame. Displacement vectors are then encoded so that the decoder can reconstruct the prediction of the current frame from the previously decoded frame(s). As the block-based prediction is not accurate enough to encode perfectly the current frame, the error between the original and the predicted frame is encoded separately. This is in general referred to as texture coding or motion residual coding. The main drawback of this method lies in the blocky nature of the prediction mechanism, which gives rise to very noticeable blocky artefacts at low bit rates. Moreover such a system, while well suited for wide translational motions, is unable to cope with locally complex movements or even global geometric transformations such as zoom or rotation. Finally, block based motion prediction is not able to follow natural features of images since it is stuck in a fixed framework based on artificial image primitives (blocks).

This invention propose to solve the above-mentioned problems by introducing a new paradigm for dealing with spatio-temporal redundancy.

The method according the invention is defined in claim 1. In the dependent claims various embodiments are proposed.

The invention will be described with the help of accompanying representations.

Figure 1: shows the progressive reconstruction of an I frame with 50, 100, 150 and 200 atoms.

Figure 2 : shows a flow-chart of the I-frames codec, where the atoms are used to transmit the frame.

Figure 3: shows a flow-chart of the I-frames codec, where the atoms are estimated from the coded image both at the encoder and decode.

Figure 4 : shows three successive schematic updates of basis functions inside a sequence of frames

Figure 5 : shows a flow-chart of the P-frames coder, using atom based motion prediction.

Figure 6 and 7: show the encoding of the Foreman sequence: I-frame, predicted frames #20, 50 and 99.

Figure 8: represents PSNR along the encoded Foreman sequence as a function of the number of predicted frames.

The proposed method is composed of two main parts. In a first step, a geometric model of a reference frame is built. In the second step, the model is updated by deformations in order to match successive frames.

A reference frame (or intra frame, I-frame) is first decomposed into a linear combination of basis functions (atoms) selected in a redundant, structured library (see . P. Vandergheynst and P. Frossard, Efficient image representation by anisotropic refinement in matching pursuit, in Proceedings of IEEE ICASSP, Salt Lake City UT, May 2001, vol. 3, the content of which is incorporated herein by reference)

$$I(x,y)=\sum_{n=0}^{N-1} c_n g_{\gamma_n}(x,y),$$

In this equation, *I(x,y)* is the Intensity of the I-frame represented as a function giving the gray level value of the pixel at position (x,y). c_n are weighting coefficients and $g_{\gamma_n}(x,y)$ are the atoms involved in the decomposition. These atoms are image patches generated by a simple mathematical formula expressing the gray level value at pixel position *(x,y)*. The formula is built by applying geometric transformations to a function *g(x,y)* that we call a generating mother function. The parameters of these transformations are concatenated in the vector of parameters γ. Examples of possible transformations are translations, rotations or dilations. They act on the generating mother function by change of variable, for example:

Translations: $g_b(x,y) = g(x-b_1,y-b_2)$

Dilation: $g_a(x,y) = a^{-1}\, g(x/a,y/a)$

Translations, anisotropic dilations and rotations:

$$g_{\gamma}(x,y) = g(x_n, y_n), \quad \text{where} \quad \begin{array}{l} x_n = \dfrac{\cos\vartheta(x-b_1)-\sin\vartheta(y-b_2)}{a_1} \\ y_n = \dfrac{\sin\vartheta(x-b_1)+\cos\vartheta(y-b_2)}{a_2}, \end{array}$$

and $\gamma=[a_1,a_2,b_1,b_2,\vartheta]$, are the parameters of this transformation.

Generating mother functions are chosen almost arbitrarily. Their properties can be adapted to the specific application. A possible example is to select an oscillating function of the form:

$$g(x,y) = (1-x^2)\exp\left(\frac{x^2+y^2}{2}\right)$$

The decomposition can for example be accomplished using a Matching Pursuit (see S. Mallat and Z. Zhang, Matching Pursuits with time-frequency dictionaries, IEEE Transactions on Signal Processing, 41(12):3397-3415, December 1993, the content of which is incorporated herein by reference). Matching Pursuit (MP) is a greedy algorithm that iteratively decomposes the image using the following scheme. First the atom that best matches the image is searched by maximizing the scalar product between the image and the dictionary $\langle I|g_{\gamma}\rangle$, and a residual image is computed:

$$I=\langle I|g_{\gamma_0}\rangle g_{\gamma_0} + R_1.$$

Then the same process is applied to the residual:

$$R_1=\langle R_1|g_{\gamma_1}\rangle g_{\gamma_1} + R_2,$$

and iteratively:

$$R_n = \langle R_n | g_{\gamma_n} \rangle g_{\gamma_n} + R_{n+1}.$$

Finally this yields a decomposition of the image in terms of a sum of atoms:

$$I(\mathbf{x}) = \sum_{n=0}^{N-1} \langle R_n | g_{\gamma_n} \rangle g_{\gamma_n}(\mathbf{x}) + R_N(\mathbf{x}).$$

The basis functions (atoms) are indexed by a string of parameters γ_n representing geometric transformations applied to a generating mother function *g(x,y)*. This index can be seen as a point on a manifold. The set of geometric transformations is designed in such a way that the total collection of basis functions (atoms) is a dense subspace of $L^2(R^2)$, i.e. any image can be exactly represented.

This part of the method expresses the I-frame as a collection of atoms that can be seen as geometric features such as edges or parts of objects which are very noticeable by the human eye. These basic primitives hereafter referred to as atoms, form a primal sketch of the image. The atoms are modelled and fully represented by the set of coefficients and parameters $\{c_n,\ \gamma_n,\ n = 0,\ldots,N-1\}$, where c_n is the coefficient and γ_n is a vector of parameters.

There are two ways to handle the coding of the I-frames. The first one is to estimate the atoms of the original frame. The atoms modelling the I frame are then quantized, entropy coded and sent in the bitstream. The process of quantizing an atom corresponds to the quantization of its coefficient and parameters. The atoms are also stored in memory in order to be used for the prediction of the next frames. A flowchart of this procedure is shown in Figure 2, where the atoms are used to transmit the frame. Dotted lines represent the coefficients and parameters of the atoms, the bold solid line represents a video frame and the light solid line represents bitstreams. The flow-chart should be understood as follow: The input frame is decomposed into atom using the MP algorithm. The atoms are then quantized and entropy. They are also de-quantized and stored in memory for the prediction of the next frames. Those de-quantized atom parameters are used to reconstruct the image, with the help of the generating mother function (which is known both by the encoder and decoder). Finally the difference between the original frame and the reconstructed one is computed and encoder using the frame coder.

Note that the figure includes an optional step, that encodes the motion residuals (or texture), which is the difference between the original frame and the one reconstructed using the atoms. This encoding can be used to further increase the quality of the decoded image up to a lossless reconstruction.

The second way of handling the I-frames is more conventional. The original frame is encoded and transmitted using any frame codec. Then the atoms are estimated from the reconstructed frame, both at the encoder and at the decoder. Finally those atoms are stored in memory for the prediction of future frames. The flowchart of this procedure is shown in Figure 3, where the atoms are estimated from the coded image both at the encoder and decoder. Dotted lines represent the coefficients and parameters of the atoms, the bold solid line represents a video frame and the light solid line represents bitstreams. The flow-chart should be understood as follow: The input frame is first encoder with the frame coder and send to the decoder in a bitstream. It is also decoded, in order to get the exact same frame that will be available at the decoder. This frame is decomposed into atoms by the MP algorithm. Finally those atoms are stored in memory to be used for the prediction of the next frames.

The second step of the method consists in updating the image model (the set of all atoms) in order to take into account the geometric deformations that have occurred between the reference and the current frame. Clearly, since the model is based on geometric transformations, updating its atoms allows for adapting to smooth local distortions (translations, rotations, scales are common examples). In order to compute this update, we assume that the deformed model is close enough to the reference model. We thus have to search for new atoms parameters in the proximity of the previous solution. This is performed by means of a local optimization procedure trying to minimize the mean square error between the updated model and the current frame (Figure 4, where three successive schematic updates of basis functions (atoms) inside a sequence of frames are represented). The updated atom parameters are then the solution of:

$$\underset{\{c_n,\gamma_n\}}{\mathrm{ArgMin}}\left\{\left\|\sum_n c_n g_{\gamma_n} - I_t\right\|\right\},$$

where the optimization method for frame I_t at time t is initialized with the atom parameters corresponding to the solution at time $t-1$ or to the reference frame (the I frame) in order to avoid error propagation. This problem is a non-convex, non-linear, differentiable optimization problem (see Dimitri P. Bertsekas (1999) Nonlinear Programming: 2nd Edition. Athena Scientific http://www.athenasc.com/nonlinbook.html, , the content of which is incorporated herein by reference), which can be solved using various algorithms such as quasi-Newton methods, combined with line-search or trust-region globalization techniques (see Conn A., Gould N. & Toint Ph. (2000) Trust Region Methods. SIAM. http://www.fundp.ac.be/~phtoint/pht/trbook.html, the content of which is incorporated herein by reference), in order to identify a local optimum.

The difference between the original atom parameters and the updated ones is then computed and sent together with updated atom coefficients. Quantization and entropy coding can then be performed (P. Frossard, P. Vandergheynst and R.M, Figueras y Ventura, Redundancy driven a posteriori matching pursuit quantization. ITS Technical Report 2002, the content of which is incorporated herein by reference) and a bit stream generated. This procedure is shown in detail in the flow-chart of Figure 5. representing a flow-chart of the P-frames coder, using atom based motion prediction. Dotted lines represent the coefficients and parameters of the atoms, the bold solid line represents a video frame and the light solid line represents bitstreams. The flow-chart should be understood as follow: The input frame are passed to the parameter estimation, which will modify the atom parameters stored in memory in order for them to describe correctly the input. The difference between the new atom parameters and the one stored in memory is computed and the result is quantized and entropy coded. Those quantized difference are also de-quantized and added to the atom parameters previously stored in memory. This allows the reconstruction of the same atoms as the one available at the decoder. Those reconstructed atom parameters are then stored in memory, replacing the ones of the previous frame. They are also used to reconstruct the current frame using the generating mother function available at the encoder and decoder. The difference between this reconstruction and the original input is computed and encoded using the frame coder. The number of updated atoms and their quantization can be fixed but can also be chosen in adaptive manner through rate and distortion constraints by means of a rate controller. For each motion predicted frame, the motion residuals (or texture) can be computed and encoded using any still image codec. It would then be sent in the bitstream in order to generate a scalable stream achieving lossy to lossless compression.

Typical results of the motion prediction, with a coding cost of 170 Kbps in average are shown in Figure 6, where predicted frames 20, 50 and 99 are represented.

It can be seen that the motion prediction stays accurate even after 100 frames, even in the absence of encoding of the motion residual. This is the major advantage of this technique compared to block based compensation. In order to show the advantage of the new prediction technique, we have done the same experiment with typical block matching compensation. The same sequence was encoded using adaptive block compensation with block sizes of 4x4 to 32x32. The encoder automatically selected the best block size according to Rate-Distortion optimisation. By varying the block size it is possible to control the size of the compressed bitstream in order to match the result of the atom based motion prediction. Like in the case of the atom based prediction, the motion residual where not coded. The Motion prediction results of the block matching are shown in Figure 7.

Objective comparison can be done using the PSNR measure. This measure is computed using the squared difference between the original and the reconstructed frame, i.e. $PSNR = -10\log\left(\frac{255^2}{\sum_{x,y}(I(x,y) - I_r(x,y))^2}\right)$, where *I(x,y)* is the original frame and $I_r(x,y)$ is the reconstructed frame. The PSNR comparisons of the two prediction methods, for a bitstream of average size of 170 Kbps, are shown in Figure 8. The performance of the current atom based motion prediction is particularly good taken into consideration, that the states of the art block based motion prediction was used. Moreover, it can be seen that in the long term (more than 50 frames) the prediction of the atom based method is more constant. In typical block matching applications, the block size is limited to 8x8, which causes poorer performances for the Block Matching.

The number of predicted frames can be fixed but can also be chosen in adaptive manner through rate and distortion constraints by means of a rate controller. For instance, when abrupt changes occur (shots between scenes), the model is not an accurate base anymore and the reference I-frame can simply be refreshed in order to compute a new model. Such a refresh mechanism can be monitored by tracking frame-to-frame distortion, which should stay rather constant for smooth updates of the initial model (Figure 8).